Praktikumsarbeit DSSN WS 2014: Statistik der Xodx-Simulation

Franz Teichmann

26. März 2015

Inhaltsverzeichnis

1 Einleitung und Aufgabenstellung

Im diesjährigen DSSN-Praktikum (Xodx) ging es im Allgemeinen um die Simulation eines Verteilten Sozialen Netzwerkes nach dem Architekturkonzept von Sebastian Tramp et al.[2], in welchem semantische Technologien zum Einsatz kommen. Im Team von 5, später 3 Studenten war es unsere Aufgabe, mehrere Instanzen der Xodx-Software[1] auf Docker-Containern[2] aufzusetzen und unter Verwendung eines Aktivitätskorpus bestehend aus Twitterdaten diese, nachfolgend Agenten genannten Instanzen, miteinander kommunizieren zu lassen.
Es sollten dabei typische Aktivitäten in einem sozialen Netzwerk wie das Teilen von Inhalten, das Hinzufügen und Entfernen von Freunden usw. simuliert werden. Anschließend sollte die Funktionalität, Korrektheit und Belastbarkeit dieser Software ausgewertet werden. Dabei ging es bewusst nicht darum, Eigenschaften des sozialen Netzwerkes und damit die Qualität oder Repräsentativität der Simulation zu analysieren, sondern vielmehr um die Feststellung von funktionalen Mängeln in der Versuchsanordnung.
Meine Aufgabe im Praktikum war es, ein Framework zu finden und vorzubereiten, mit dem diese Messungen erfasst und evaluiert werden können. Dabei soll auf folgende Fehlerklassen geprüft werden:

1. Überprüfung der Dateneffizienz des Xodx-Netzwerkes - ,,Triple-Explosion"
   Es ist anzunehmen, dass im Verlauf der Simulation, also mit steigender Anzahl an in den Knoten gespeicherten Nachrichten bzw. Beiträgen anderer Agenten, der lokale Speicherverbrauch jedes Agenten steigt. Dabei ist abhängig vom Versuchsaufbau mit einem linearen oder mit einem gleichmäßigen polynomiellen Wachstum zu rechnen. Sollte der Speicherverbrauch exponentiell oder unregelmäßig an einem bestimmten Punkt in der Simulation nach oben schnellen, ist dies ein Hinweis auf einen potentiellen Fehler des verwendeten Triplestores.

2. Überprüfung auf verlorene Nachrichten
   Da wir das System von Twitter adaptieren, in dem Nutzer andere Nutzer abonnieren und somit nach dem Paradigma des Broadcast alle zukünftigen Nachrichten dieses abonnierten Nutzers erhalten, muss die eingehende Zahl an Nachrichten im gesamten Netz gleich der ausgehenden Nachrichten mal Anzahl der ,,Follower" jedes Agenten sein. Sollten diese Werte nicht übereinstimmen, ist das ein Zeichen für verlorene Nachrichten.

3. Zugriffsgeschwindigkeit auf die eigene Zeitleiste
   Die Zeitleiste ist die Übersicht aller empfangener Nachrichten und kann nach einem Login im Account eingesehen werden. Die Zugriffszeit darauf ist ein Indiz für die Effizienz der Software. Ähnlich Punkt 1 ist eine Unregelmäßigkeit bzw. ein sprunghafter Anstieg der Zugriffszeit ein Zeichen für einen Fehler zur Laufzeit.

[1] Xodx Projektseite im AKSW - `http://aksw.org/Projects/Xodx.html`
[2] Docker - eine freie Software zur portablen Verwendung von Apps in virtuellen Maschinen `https://www.docker.com/`

Die Ergebnisse der Recherche nach einem passenden Statistikframework und die Umsetzung dessen sollen in den kommenden Kapiteln vorgestellt werden.

2 Recherche

Die Darstellung von Statistik im Netz ist eine schwierige Aufgabe, denn Statistik ist stets nur eine Sicht auf vielschichtige und potentiell höherdimensionale Daten. Sie kann daher nur einen Ausschnitt der gemessenen Wirklichkeit darstellen. Dies entsteht mithilfe von statistischen Mitteln zur Verdichtung der vorhandenen Information unter Ansteigen der Entropie und geht mit mehr oder weniger gravierendem Informationsverlust einher, der beim Herunterbrechen auf eine darstellbare Dimensionalität der Daten nur schwer zu minimalisieren ist.
Folglich ging es im Praktikum darum, Kenngrößen zu ermitteln und deren Messwerte statistisch so zu erfassen, dass sie unter möglichst geringem Informationsverlust ausgewertet und in einer ansprechenden Form dargestellt werden können.
Weiterhin sollte ein Statistikframework verwendet werden, welches das Konzept der Erweiterbarkeit beinhaltet, da ausgehend von den einzelnen Agenten der Simulation viele Einzelmesswerte auftreten und auch mehrere Testläufe geplant sind.
Die Agenten kommunizieren untereinander bereits in RDF, weshalb sich aus Integritätsfragen ein Framework aus der Linked Data Welt anbietet. Dieses würde mit den Eigenschaften von RDF auch die Forderung nach der Erweiterbarkeit erfüllen.
Das bekannteste Statistikvokabular für Linked Data heißt Data Cube und hat sogar eine W3C Empfehlung[3] bekommen. Es soll in Kapitel 3 kurz vorgestellt werden.

3 Das Data Cube-Vokabular

Das Data Cube Vokabular zielt darauf ab, unter Einhaltung des RDF Standards die Publikation von Statistik oder anderen höherdimensionalen Daten zu ermöglichen. Es baut dabei auf SDMX[3] auf, einem Standard für den Austausch von Statistischen Daten im Unternehmenskontext, welcher 2013 in den Status eines ISO Standards[4] gehoben wurde. Es sollen zunächst die Modellstruktur und die Datentypen vorgestellt werden, um anschließend ausgehend von den Komponenten ein Beispiel zu erläutern.

3.1 Data Cube Modellstruktur

In Abbildung 1 ist die Modellstruktur des Data Cube Vokabulars zu sehen. Unter dem Kürzel qb steht der Namespace `http://purl.org/linked-data/cube#`. Zentrale Komponenten sind mit den Messwerten die qb:observations, welche an

[3]SDMX: Statistical Data and Metadata Exchange `http://sdmx.org/`

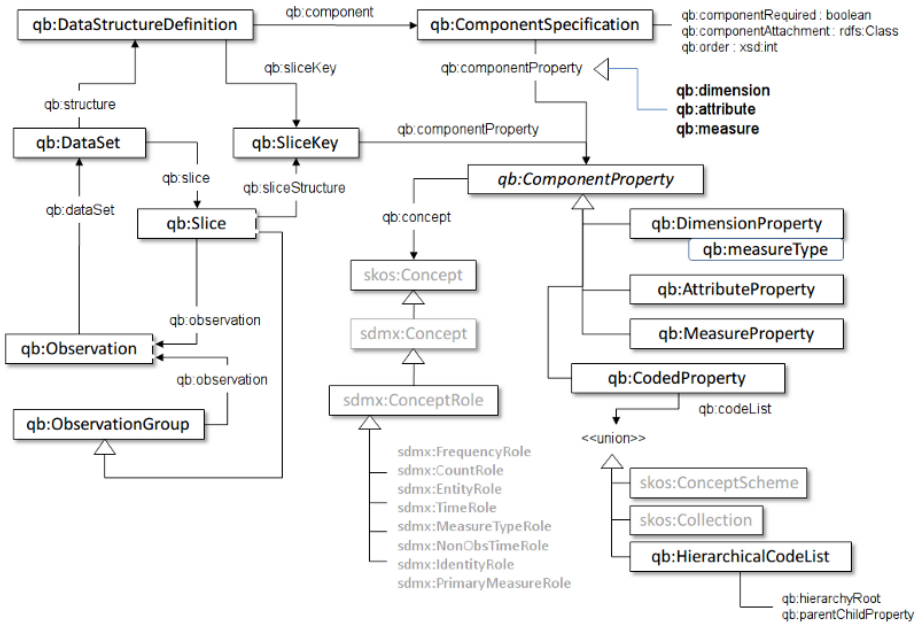

Abbildung 1: Outline des Data Cube Vokabulars

ein DataSet gekoppelt sein müssen. Zu jedem DataSet gibt es eine Strukturdefinition in Form einer DataStructureDefinition, welche wiederum eine Anzahl an Komponenten deklariert, aus denen das DataSet besteht. Diese Komponenten haben je nach Typ Komponenteneigenschaften (ComponentProperty), auf welche in Kapitel 3.3 eingegangen werden soll. Des weiteren gibt es Gruppierungen innerhalb eines DataSets in Form von sog. qb:Slices stelltvertretend als Begriff für die reduzierte Betrachtungsdimension auf eine Statistik. Ausgehend von diesen Betrachtungsdimensionen sollen nun die Datentypen des Vokabulars vorgestellt werden.

3.2 Datentypen

Alle Messwerte einer Statistik sind auf einen vorher definierten logischen Raum bezogen, dessen Dimensionen gleichzusetzen sind mit den einzelnen Messgrößen. Sie kann mit der mathematischen Struktur des Hypercubes verglichen werden, daher auch der Name Data Cube. Zur Darstellung dieser dimensionalen Daten sowie einer auf einer Metaebene stattfindende Beschreibung werden vier Datentypen definiert:

1. Observations
   Dies sind alle Beobachtungen, welche im logischen Raum der Statistik getätigt wurden. Umgangssprachlich ist all das eine Observation, dessen

Wert durch das ,,Anlegen des Lineals" ermittelt wurde. Sie sind vergleichbar mit den literalen Werten in den Zellen einer Messwertetabelle.

2. Organizational Structure
   Die den Observations zugrunde liegende organisatorische Struktur wird in Form von Dimensionsscheiben ausgedrückt. Wenn die Observations die Tabellenwerte sind, so wäre die organisatorische Struktur die Beschriftung der Zeilen und Spalten der Messwertetabelle.

3. Structural Metadata
   Die Angabe von Metadaten zu einer Struktur ist eine der größten Stärken von RDF. So ist es problemlos möglich, mit den Mitteln des Data Cube Vokabulars z.B. Angaben zur Methode der Messung, Maßeinheiten, Datum usw. auf verschiedenen Ebenen hinzuzufügen.

4. Reference Metadata
   Zuletzt können, um dem Design des Linked Data Konzeptes gerecht zu werden, noch Metadaten in Form von Referenzen zu anderen Wissensbasen hinzugefügt werden. Es werden dabei Beschreibungen zum Data Set in seiner Gesamtheit vorgenommen. So kann zum Beispiel die Datenquelle verlinkt oder eine Form der Kategorisierung für die Statistik angefügt werden.

3.3 Komponenten

Diese gennanten Datentypen sind, abgesehen von den Referenzdaten, strikt voneinander getrennt auf die drei Komponenten einer Observation innerhalb des Data Cubes aufgeteilt:

1. Dimensions
   Wie bereits genannt, sind die Dimensionen einer Statistik mit den Zeilen- und Spaltenbeschriftungen einer Messwertetabelle assoziiert. So kann eine einzelne Observation eindeutig durch die Angabe der Werte aller Dimensionen identifiziert werden.

2. Measures
   Zusätzlich zu den Dimensionswerten folgt eine Observation stets einer exakt definierten Messgröße, diese wird unter der Komponente Measure beschrieben und enthält damit zum Beispiel gemessene Werte wie Alter oder Körpergröße.

3. Attributes
   Zuletzt werden unter Attributes neben dem literalen Wert der Beobachtung noch Angaben zur Art und Weise der Bestimmung der Messgröße gemacht. Dazu gehören zum Beispiel die Maßeinheit und der Zeitpunkt der Messung.

Nur die korrekte und vollständige Angabe aller Komponenten einer Messung ermöglicht eine stringente Auswertung des Data Cubes und einen sinnvollen Schnitt durch die Dimensionen der Statistik, um eine auswertbare Abbildung zu generieren.

3.4 Beispiel

Eine Beispielstatistik als Anwendung für das Vokabular ist auf den Seiten des W3C[4][5] gegeben. Siehe dazu Tabelle 1; es wurden Lebenserwartungen in Wales für Männer und Frauen in drei Zeiträumen und vier Orten untersucht. Dies bedeutet, dass die Statistik aus 3 Dimensionen besteht: Geschlecht, Zeitraum, Ort. Es wird nun mit einem Data Set begonnen, in welchem Titel, Beschreibung, Autoren und nicht zuletzt auch alle Data Structure Definitions zusammengefasst sind. Es heißt im Beispiel aus Tabelle 1 ,,Life Expectancy".
In diesen, dem Data Set untergeordneten, Data Structure Definitions sind alle Dimensionen, ihre Ordnung untereinander und alle weiteren Komponenten wie in Kapitel 3.3 definiert. Im Allgemeinen ist für die Darstellung einer Messgröße je eine Data Structure Definition notwendig, also im W3C Beispiel nur eine Einzige.
Dabei werden wie beschrieben die Dimensionen und Measures verlinkt, die auch jeweils eine Strukturdefinition in Form einer Dimension Property bzw. einer Measure Property für die Definition und korrekte Verlinkung besitzen.
Zuletzt folgt eine lange Liste an einzelnen Observations, die alle eindeutig dem Data Set zugeordnet sind und für jede Dimension einen Wert besitzen. Sie können dem Konzept des Schnittes durch Dimensionen folgend angeordnet werden. So sind im ,,Lebenserwartung" Beispiel zunächst alle Observations im Slice ,,male", danach alle Observations im Slice ,,female" genannt.

male	2004-2006	2005-2007	2006-2008
Newport	76.7	77.1	77.0
Cardiff	78.7	78.6	78.7
Monmouthshire	76.6	76.5	76.6
Merthyr Tydfil	75.5	75.5	74.9
female	2004-2006	2005-2007	2006-2008
Newport	80.7	80.9	81.5
Cardiff	83.3	83.7	83.4
Monmouthshire	81.3	81.5	81.7
Merthyr Tydfil	79.1	79.4	79.6

Tabelle 1: Statistik Beispiel - Lebenserwartung in Wales

Als Veranschaulichung dieser Herangehensweise soll Tabelle 2 als mögliche Messwertetabelle im Xodx-Projekt dienen. Es werden bei 3 Agenten zu einem

[4] `http://www.w3.org/TR/vocab-data-cube/#example`
[5] `http://www.w3.org/TR/vocab-data-cube/#full-example`

Agent	1	2	3
size (kByte)	10	8	15
follower(number)	3	8	100
activity(tweet/day)	4	3	15

Tabelle 2: Statistik Beispiel - Xodx

nicht näher beschriebenen Zeitpunkt die drei Messgrößen ,,size" für Größe des Knotens in kByte, ,,follower" für Anzahl der diesen Agenten abonnierten Knoten und ,,activity" als Maß für die Aktivität des Agenten in Tweet pro Tag erfasst. Von unten nach oben aufgebaut ist leicht abzulesen, dass für die Darstellung dieser Statistik mit Data Cube 9 Observations in den 4 Dimensionen Agent, size, follower und activity eingetragen sind. Da diese Observations jedoch nicht vierdimensional sind, denn sie stellen nur eine Beziehung zwischen Agent und Messgröße dar, sind an dieser Stelle eigentlich 3 einzelne Tabellen zusammengefasst und es werden auch drei Data Sets benötigt. Folglich sind auch 3 Data Structure Definitions zu je zwei Dimensionen, wobei die Dimension des Agenten von allen gemeinsam genutzt wird, und je einer Measure notwendig. Jede dieser Dimensionen und Measures verlangt genau eine Strukturdefinition in Form einer Measure Property bzw. Dimension Property.
Dem Beispiel folgend wurde im Praktikum ein Data Cube erstellt, der die Messungen zu den Zielstellungen aus Kapitel 1 erfasst. Dieser soll im nächsten Kapitel vorgestellt werden.

4 Xodx-Datacube

Für die Erfassung der drei Fehlerklassen in der virtuellen Testumgebung werden mehrere Messgrößen benötigt, die zunächst in einer Übersicht erklärt werden sollen, um sie Anschließend über die Data Sets zueinander in Beziehung zu setzen.

4.1 Messgrößen

In Fehlerklasse 1, der ,,Überprüfung der Dateneffizienz" des Netzwerkes müssen die Anzahl der gespeicherten Tripel eines Knotens in Relation mit dem dafür verwendeten Speicher gesetzt werden. Nur so ist es möglich, durch Vergleich des dabei entstehenden Quotienten Unregelmäßigkeiten innerhalb des Netzwerkes festzustellen. Zusätzlich soll mit dem Festhalten der in einem Knoten gespeicherten Tripel die Relation zum dabei verwendeten Speicher durch Verrechnung mit dem anderen Quotienten dieser mathematisch aussagekräftiger werden. Die unter dem Präfix xo[6] definierten Data Sets sollen auf der nächsten Seite vorgestellt werden.

[6] `http://xodx.dssn.org/`

1. xo:dataset-xoIN
   Ein Data Set für die Erfassung der in einen Knoten eingegangenen Nachrichten. Diese werden im lokalen Speicher für die Timeline gespiegelt.

2. xo:dataset-xoTriples
   Die Anzahl der in einem Knoten abgelegten Tripel.

3. xo:dataset-xoSize
   Misst die Größe eines Knotens in kByte. Die Implementierung der Funktion für diese Messung muss noch im weiteren Verlauf des Praktikums im nächsten Wintersemester hinzugefügt werden, sollte aber kein Problem darstellen.

Für Fehlerklasse 2, ,,Überprüfung auf verlorene Nachrichten" sind 3 Data Sets notwendig:

1. xo:dataset-xoFollower
   Bezeichnet die Follower, die ein Agent hat.

2. xo:dataset-xoOUT
   Ist das Data Set für die Erfassung aller aus einem Knoten ausgehenden Nachrichten. Das Produkt aus ausgehenden Nachrichten und Followern muss gleich der globalen Summe aller eingehenden Nachrichten sein.

3. xo:dataset-xoIN
   Es wird das gleiche Data Set wie oben verwendet.

Zuletzt soll Fehlerklasse 3 abgefangen bzw. ausgewertet werden, indem die Zugriffsgeschwindigkeit auf die eigene Zeitleiste im Data Set xo:dataset-xoAccess erfasst wird und sich so über viele Zeitscheiben hinweg und über die Relation mit dem Data Set xo:dataset-xoTriples bzw. xo:dataset-xoSize vergleichen lässt.

4.2 Komponenten und Struktur

Es gibt nun mit xo:CSrefAgent, xo:CSrefTime und dem xo:CSvalue als Platzhalter für einen Messwert drei Komponenten, die über alle Data Sets übergreifend verwendet werden. Zusätzlich dazu hat jedes Data Set noch eine eigene Measure Property und abgesehen von ,,normalen" Ziffern einen eigene Maßeinheit, welche als Attribut angehängt wird.

4.3 Observations

Jede Observation ist dreidimensional; sie hat eine ID für den Agenten, einen Messzeitpunkt in Sekunden ab Simulationsbeginn und jeweils den Messwert als Platzhalter verknüpft mit xo:value, der Instanz von xo:CSvalue. Über den Kontext des verlinkten Data Sets ergibt sich dieser Wert des xo:value semantisch in der Maßeinheit des Data Sets. Zusätzlich dazu muss jede Observation die Agenten übergreifend eindeutig identifizierbar sein. Es reicht also nicht aus, die Agenten selbst ihre Observations benennen zu lassen, sondern es muss einen Mechanismus geben, der global Kollisionen in der Benennung vorbeugt.

5 Darstellung und Auswertung des Data Cube

Für die grafische Darstellung bzw. Auswertung eines fertigen Data Cube gibt es am AKSW das Projekt CubeViz[7]. Es handelt sich dabei um einen Datenbrowser, der unter Verwendung von RDF und dem Data Cube Vokabular automatische Grafiken und damit auswertende Sichten auf eine Statistik generieren kann. Er passt damit gut in das DSSN Projekt und eignet sich dafür, alle gesammelten Beobachtungen in Form eines großen Data Cube zusammenzuführen und anschließend zentral auszuwerten. Das Setup und die notwendigen Vorbereitungen zum Einlesen eines Data Cube in CubeViz sollen in den kommenden Kapiteln präsentiert werden.

5.1 Setup von CubeViz

Für den Start von CubeViz benötigt man, wie auf den Wikiseiten[1] beschrieben, einen lokalen Webserver auf idealerweise einem auf Linux basierenden Betriebssystem, z.B. Apache oder Nginx, auf dem folgende Dinge installiert sein müssen:

1. Virtuoso oder SQL Server
   Virtuoso und SQL sind Datenbankserver, wobei Virtuoso explizit für die Speicherung von RDF-Graphen konzipiert und optimiert ist. In der kostenfreien Version Openlink Virtuoso[8] bietet es genügend Kapazität und Geschwindigkeit für unser Praktikum. Es kann aus den Paketquellen für Linux installiert werden und dient als Backend für das Ontowiki.

2. Ontowiki
   Ontowiki[9] ist eine vom AKSW entwickelte Applikation zur Verwaltung und Pflege von Wissensbasen des Semantic Web. Es unterstützt mit RDF, Turtle und N3 die wichtigsten Formate, in denen solche Wissensbasen geschrieben sein können. Gleichzeitig dient es vergleichbar mit einem Framework als Basis für weitere, in Form von Erweiterungen installierte, Applikationen. Diese Eigenschaft nutzen wir, um darauf aufbauend CubeViz zu installieren und als automatisches Addon für die Auswertung des Data Cube zu verwenden.
   Dafür müssen die Dateien für CubeViz lediglich in den Extension-Ordner des Ontowiki kopiert werden bzw. der Stammordner dem Ontowiki bekannt gemacht werden. Nach einem Neustart des Webservers wurde das Plugin beim Einlesen eines validen Data Cube automatisch aktiv.

5.2 Einlesen des Data Cube

Um von CubeViz als auszuwertender Data Cube erkannt zu werden, muss die Wissensbasis bestimmte Eigenschaften erfüllen. Dafür schickt CubeViz intern eine Sparql-query an das Backend. Nur wenn diese Prüfung erfolgreich ist, wird

[7] CubeViz Projektseite - `http://aksw.org/Projects/CubeViz.html`
[8] `https://github.com/openlink/virtuoso-opensource`
[9] `https://github.com/AKSW/OntoWiki`

```
1  PREFIX qb:<http://purl.org/linked-data/cube#>
2    ASK
3    {
4      ?observation a qb:Observation .
5      ?observation qb:dataSet ?dataset .
6      ?observation ?dimension ?dimelement .
7      ?observation ?measure ?value .
8
9      ?dataset a qb:DataSet .
10     ?dataset qb:structure ?datastructuredefintion .
11
12     ?datastructuredefintion a qb:DataStructureDefinition .
13     ?datastructuredefintion qb:component ?dimensionspecification .
14     ?datastructuredefintion qb:component ?measurespecification .
15
16     ?dimensionspecification a qb:ComponentSpecification .
17     ?dimensionspecification qb:dimension ?dimension .
18
19     ?measurespecification a qb:ComponentSpecification .
20     ?measurespecification qb:measure ?measure .
21    }
```

Abbildung 2: Query - Überprüfung auf Data Cube

CubeViz für den Nutzer sichtbar. Diese Sparql-query ist in Abbildung 2 aufgeführt. Es muss mindestens eine qb:Observation in der Wissensbasis enthalten sein, welche einem qb:dataSet folgt. (Zeile 4,5) Dieses Data Set muss Data Structure Definitions folgen und aus Komponenten, darunter Dimensionen bestehen, die ebenfalls eine Spezifikation haben. Eine dieser Spezifikationen muss in Form einer Measure Property qb:measure bestehen. (Zeile 20)
Der Data Cube des Praktikums aus Kapitel 4 erfüllt diese Forderungen und wird von CubeViz erkannt. Es ist möglich, im Auswahlfeld des Plugins verschiedene Komponenten auszuwählen und mit automatisch generierten Diagrammen auszuwerten.

6 Zusammenfassung

Im diesjährigen Praktikum hatte ich die Aufgabe, ein Statistikframework zu finden, mit dem sich die Simulation des verteilten sozialen Netzwerkes auf Basis der Xodx-Software auswerten lässt. Leider waren wir im Praktikum selbst nur sehr wenige Studenten, von denen nur drei bis zum Ende blieben. Dadurch war es uns nicht möglich, das Projekt bis zu einer tatsächlichen Simulation voranzubringen.
Trotzdem konnte mit dem Data Cube Vokabular ein sehr viel versprechendes

Vokabular gefunden werden, welches zusätzlich dem Linked Data Paradigma gerecht wird und sich daher sehr gut in die Philosophie des DSSN einfügt. Nach einer gründlichen Recherche des Vokabulars wurde ein Data Cube entworfen und geschrieben, dessen Data Sets eine Analyse der Simulation auf die drei in der Planung festgestellten potentiellen Fehlerklassen erlauben und so möglicherweise schon während des Testlaufes eine aussagekräftige Auswertung ermöglichen. Die Implementation der Messfunktionen und das automatische Ablegen der Observations müssen noch in einer Fortführung des Praktikums geschrieben werden, sollten aber hinsichtlich des Schwierigkeitsgrades z.B. mit der Scriptsprache PHP[10] zu bewältigen sein.
Auch gelang unter Verwendung des CubeViz-Plugin für das Ontowiki bereits eine Auswertung des Data Cube mit einigen beispielhaften Observations. Es ist möglich, automatisch generierte Grafiken zu erzeugen, die in übersichtlicher Form die gesammelten Beobachtungen zusammenfassen und auf einen Blick eine Prüfung auf die Fehlerklassen erlauben. Damit halte ich CubeViz für optimal geeignet, um nach einem Simulationslauf die Auswertung durchzuführen.
Alle Quelldateien sind auf meinem Github-Account[11] im DSSN Projekt[12] gesammelt und werden auch weiterhin gepflegt. Es sind noch ein paar kleinere Anpassungen und eine genauere Dokumentation des Data Cube vorzunehmen, um ihn auch im weiteren Verlauf des Praktikums im nächsten Wintersemester einsetzen zu können.

Literatur

[1] Konrad Abicht. Cubeviz - the rdf datacube browser im github wiki. `https://github.com/AKSW/cubeviz.ontowiki/wiki`. Zuletzt gesehen: 03/2015.

[2] Sebastian Tramp et al. An architecture of a distributed semantic social network. `http://www.semantic-web-journal.net/sites/default/files/swj201_4.pdf`. Zuletzt gesehen: 03/2015.

[3] NUI Galway Richard Cyganiak, DERI. The rdf data cube vocabulary. `http://www.w3.org/TR/vocab-data-cube/`. Zuletzt gesehen: 03/2015.

[4] various. Iso 17369:2013. `http://www.iso.org/iso/catalogue_detail.htm?csnumber=52500`. Zuletzt gesehen: 03/2015, volles Dokument zum Standard nur kostenpflichtig erhältlich.

[10] `http://php.net/`
[11] `https://github.com/FTeichmann`
[12] `https://github.com/DSSN-Practical/DSSN_Statistics`